# Processing Web Editor for Data Processing in a Digital Oscilloscope or Similar Instrument

## Cross-Reference to Related Applications

This application claims the benefit of U.S. Provisional Application Serial No. 60/249,482 filed November 17, 2000, the entire contents thereof being incorporated herein by reference.

## Background of the Invention

In a traditional scheme for controlling a digital oscilloscope or the like, there has been no application of a clear mental or abstract generic model for processing various acquired waveforms and the like. Furthermore, the available functionality has been primarily predefined, without much flexibility on the part of the user. This is likely because very few people who have participated in the development of digital oscilloscopes, much less the users of these instruments, have had a very clear idea of how to express the processing which is performed within these instruments.

## Summary of the Invention

The inventors of this application have determined that a complete abstract and generic description is essential to leveraging the powerful mathematical tools that are present as features in modern oscilloscopes and other digital processing apparatus. In other words a proper abstract and generic description of such a processing apparatus and system is essential to forming a useful understanding of how the instrument works, how it can be used to solve problems, and how various features available in the instrument may be used in a beneficial manner for processing of signals. This is because while many longstanding leaders in the data processing and particularly oscilloscope industry provide feature-laden products, the full benefit of these features has not been yet recognized. Most users simply do not have a sufficient understanding

of what is happening inside the instrument and how to properly configure the instrument to utilize all of the advanced features.

Digital Storage Oscilloscopes (DSOs) have traditionally provided processing of a captured waveform in two ways. These two ways have included Math features and Parameter features. The math feature takes a captured waveform and processes it to produce another waveform. The Parameter feature takes a captured waveform and produces parametric measurements of the waveform. In more traditional oscilloscopes and processing apparatus, these two forms of processing are controlled by menus that separates the two concerns. Math is typically controlled by a menu that allows pre-defined, pre-named math functions to be selected (typically five of these are supported in a LeCroy oscilloscope). Parameters are typically controlled by a menu that allows for a predefinition of a number of parameter measurements. There are typically five such parameters. These menus traditionally allow a single processing element to be defined and configured per each named function. More recent Digital Oscilloscopes include more flexible menus supporting a definition of a math function algebraically, allowing the concatenation of a limited number of processing steps.

However, math and parameters are still thought of as separate entities, one producing vector results (waveforms) and the other producing scalar results (parameters). Most DSOs provide no connection between the two. For example, it is generally not possible to configure a math function to consume the results of a parameter measurement. Certain DSOs could provide some connection between the two, in the form of Histograms and Trends of parameters, but this connection is at best tenuous.

Also, other forms of processing exist in DSOs although they are rarely presented to the user. For example, the persistence of a series of waveforms is generally performed by the

display system in a DSO. However, the resulting two-dimensional persistence map is not traditionally available to the user other than visually on the display, and therefore cannot be acted upon or consumed by the DSO for further processing. The same applies, for example, to the display of two waveforms in an X vs. Y format. Thus, at this time a flexible system for visualizing and defining an entire processing flow of a DSO, from the captured waveform through to the display and/or other output devices, in a graphical manner, does not exist.

Still other objects and advantages of the invention will in part be obvious and will in part be apparent from the specification and the drawings.

The invention accordingly comprises the several steps and the relation of one or more of such steps with respect to each of the others, and the apparatus embodying features of the construction, combinations of elements and arrangement of parts that are adapted to effect such steps, all as exemplified in the following detailed disclosure, and the scope of the invention will be indicated in the claims.

## BRIEF DESCRIPTION OF THE DRAWINGS

For a more complete understanding of the invention, reference is made to the following description and accompanying drawings, in which:

Fig. 1 is a graphical representation of a processing web for a LeCroy, traditional, fixed-feature instrument in accordance with the invention;

Fig. 2 depicts various examples of processor classes;

Fig. 3 depicts an example of an advanced processor;

Fig. 4 depicts a simple processing web configuration;

Fig. 5 depicts a further processing web configuration;

Fig. 6 depicts separate processing paths in a processing web;

Figs. 7A and 7B are further graphical representations of processing webs in accordance with the invention;

Fig. 8 depicts a plurality of miniature output renditions;

Fig. 9 depicts parameters that may be set for a histogram node;

Figs. 10A-10C depict a sequence for connecting two processing nodes in which an adapter is automatically inserted;

Figs. 11A-11C depict a further sequence for connecting two processing nodes in which an adapter is automatically inserted;

Fig. 12 depicts a collection of components available in a Math category;

Fig. 13 depicts a simple form of a processing web;

Fig. 14 depicts a further form of a processing web;

Fig. 15 depicts a further form of a processing web;

Fig. 16 is a flowchart depicting a sequence for populating a processing web editor screen in accordance with an existing processing web;

Fig. 17 is a flowchart depicting a sequence for dropping a new component into a processing web using a processing web editor;

Fig. 18 is a flowchart depicting a sequence for attempting to connect two pins in a processing web using a processing web editor;

Fig. 19 is a view of a functioning processing web editor;

Fig. 20 is a further view of a functioning processing web editor; and

Fig. 21 is a depiction of a conceptualization of the operation of an oscilloscope constructed in accordance with the invention.

## DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

In accordance with the invention, an oscilloscope design is provided that is completely "modular", whereby the elements composing the oscilloscope software are completely specified in terms of interface, and are sufficiently independent in construction that they can be installed and removed at run-time, (e.g. without stopping the process which can be considered the oscilloscope application software). This is in contrast to existing products that include software that must be wholly replaced (in it's entirety) in order to change any small attribute of the software. This inventive approach offers commercial advantage because is shortens the overall logistics of releasing new software features, fundamental to the product or fundamental to a particular application of the product. As such, a special feature can be added after a customer has taken delivery and has the instrument (oscilloscope) in operation, after the fact, without the requirement to return or for that matter even stop using the instrument. This is of value to both the customer and the manufacturer.

Further, the modular design permits "unit testing" to further advance the quality of the overall product by permitting each individual component of the software to be tested in a standard "test harness". Each component is provided with a standardized interface especially conceived to permit isolated testing. This forward looking, quality conscious aspect of the design assures final deployments of the product(s) are not hampered by the inability to localize faulty components.

Fig. 21 depicts a conceptualized notion of how the oscilloscope constructed in accordance with the invention operates. The flow of data is shown from left to right and control sequences provided by a user, either automated or manual, are shown from top down. As is shown in Fig. 21, a plurality of analog acquisition signals $C_1$ are acquired by the oscilloscope of

the invention. These acquisition signals are acquired in accordance with various provided probes and other acquisition hardware 2110 that are in turn controlled by an acquisition control module 2112. Various acquisition control processes are applied to the acquired waveforms in accordance with various control variables 2120 provided by the user via a user interface 2115. These control variables control acquisition control module 2112 as well as all other oscilloscope functioning to be discussed below.

Various results data are then provided to, and thereafter output from a preprocessing system 2121, and are in turn acted upon in various post processing functions 2130 employing user defined control variables 2120, resulting in processed data (Results data) 2135. The post-processed data is then exported and/or imported at 2140 as necessary for further processing by the system 2145, also in accordance with control variables 2120. After processing has been completed, the various processed data is converted for display at 2150 on a display device 2155.

A fundamental difference between this conceptual model and previous oscilloscope models is at least the freedom permitted in the section described as "processing". In most instruments prior to this design there has always been a fixed (and limited) number and configuration of processing results. This new oscilloscope design in accordance with the invention employs a "processing web". Various implementations and the overall structure of the processing web will be described below.

Referring next to Fig. 1, a graphical representation of a processing web representation of a fundamentally inflexible oscilloscope (older) processing design is shown. While the underlying structure of the oscilloscope is inflexible and is well known in the art, the graphical processing web depiction of the invention may still be applied. As will be described below, it is this fundamental graphical processing web description of the invention that allows the further

flexible and infinitely configurable hardware of the invention, to be constructed in accordance with the invention. It should also be noted that while the processing web is depicted in a graphical manner, the web is in fact a definition of interrelationships between processing elements, and the control of their overall interaction. Therefore, this graphical representation, while useful, is not in fact a part of the processing web.

As is shown in Fig. 1, a processing web 100 first comprises data sources of dynamic (channel) data 110. These dynamic data sources typically include various waveform acquisitions. Also included are static (memory) data types 120 representing stored input data. In accordance with the invention these input modes are usually positioned at the left in a "left-to-right" graphical representation. Further shown in Fig. 1 are a plurality of output views and measurable objects at the right edge of the web picture. Connections from data sources 110 to views 130 is direct as is shown by the direct line between them and the one-to-one relationship without any intermediate processing. Thus, these dynamic data sources are shown as being displayed without modification. Parameter settings 150 are also provided and indicate a particular parameter of one or more of the acquired waveforms that is to be measured. Also shown in Fig. 1 are trace views 140. Each trace view may receive a dynamic input 110, a stored input 120, input from a trace view, or input from a parameter 150 and perform some processing before display.

Each processing object receives certain inputs (according to various input requirements) and generates particular output types (e.g. parameters, persistence maps or waveforms). These processing objects also have particular update or synchronization characteristics for obtaining or generating information, and can be expressed using a same generic processor model, but being designated according to a variable number and type of input, output and update pins.

In this simple, fixed oscilloscope apparatus, all objects, either input or output, are represented as objects on the border of the web diagram, while the connections between the various objects are represented as connections within the processing web. The placement of these processing elements along the outer border is merely a graphical representation convention for placing inputs on one edge and outputs on another. However, this graphical display has no affect on a working system. This particular processing web graphic is a representation of an older oscilloscope design where each processing element receives predefined inputs as a combination of dynamic inputs, static inputs, parameters, and in some cases prior trace views and all function on a single system clock, all at the same speed and refresh rate. Thus, in this particular representation, there is no provision for the chaining of various processing elements, or functioning processing elements at different speeds, and indeed each processing element shown in Fig. 1 produces a predetermined output at a predetermined time. It is therefore not possible to modify the processing of the oscilloscope to provide different desirable outputs. While other LeCroy oscilloscopes have allowed for certain chaining and different processing speeds within a system. However, the ability to utilize these features was limited to a number of predetermined scenarios. As will be noted below, in accordance with the invention, these limitations have been removed, and a more general solution is provided that allows for a completely dynamic set of scenarios with differing data rates and differing requirements on different chains of processors.

In accordance with the inventive graphical representation of the processing web of Fig. 1, a Processing Web Editor (PWEditor) is a tool provided in accordance with the invention for presenting the graphical representation of the configuration of a Processing Web (Web) to a user, enabling the web to be reconfigured, and enabling the properties of the various elements in the web to be viewed and modified by the user. Thus, instead of the fixed and constrained

processing web shown in Fig. 1, a processing web for a more advanced oscilloscope may be constructed in accordance with the invention. This more advanced processing web may be edited in accordance with the PWEditor and may be configured in any manner desired by a user to generate any desired results, as will be described below. The Processing Web defines the flow of data from the input of a DSO through various stages of processing to the display device, as noted above.

While the PWEditor is the most natural way to view and reconfigure the web, it is by no means the only way. The web can also be constrained and can emulate the processing flow in conventional DSOs. It can also be configured in a more flexible manner using an equation, i.e. 'Math1 = Ch1 + Ch2'. While these possibilities are intended to be within the scope of the invention, they will not be discussed in detail herein.

Thus, the use of the PWEditor relies on the generic abstract model of the type shown in Fig. 1, but will typically be far more complicated than the web disclosed in Fig. 1 (see Figs. 7A, 7B below). In accordance with such editing, any processing object can be described via its input, output and update requirements and capabilities. Thus, it is possible to interconnect virtually unlimited numbers and types of processing objects to provide simple as well as complex processing configurations while retaining well-controlled synchronization of processing results.

There are several terms that will be used in this description that will now be defined. A Result Processor (or simply Processor) is a software object that consumes and/or produces processing Results, Examples of Results are waveforms, parameters, persistence maps, histograms, etc. In order to be useful, a Processor is typically created and inserted into a Processing Web and connected to other Processors. In other words, a Processing Web configuration is fundamentally defined by the set of Processors and their inter-connections. A

Processing Web Manager software object is responsible for managing the Processing Web and updating subsets of Processor objects in response to requests and events from other software objects.

As noted above, old software means and manners for controlling instruments are embodied in various legacy software, typically written between 1984 and now, from any number of manufacturers. In these older systems, as noted with respect to Fig. 1, there are typically a fixed number of processing objects and several different models employed for inter-connecting and updating these processing objects. These systems run on essentially a single software thread of execution, and may not be modified in any substantial manner to change the processing flow.

The drawbacks of these older, fixed systems are as follows:

1. They do not have a common, generic abstract model for inter-connecting and updating processing objects. It is therefore not possible to implement many processing combinations, let alone modify any of the existing processing elements.
2. There are typically a limited, fixed number of processing objects (e.g. 4 'math' functions, 5 'parameters', 4 'memories', etc...).
3. There are typically a limited number of processing objects that can be chained together to form a composite processing function. Further, the objects that can be chained together are predetermined, and cannot be modified by a user.
4. There are limited types of processing objects that can be implemented. Legacy math functions can have 1 or 2 inputs and produce exactly 1 result. Legacy parameter calculators can have 1 or 2 inputs and produce exactly 1 result.
5. The systems are limited by the fact that a single software thread of execution must try to handle many different system concerns with vastly different requirements (processing vs.

display-update vs. user interface response vs. etc...). The practical side effect of this is that very few processing throughput performance enhancements are available.

Therefore, in accordance with the invention, a processing web is provided, and a processing web editor for modifying the configuration of the processing web is also provided. In such a processing web model, a number of rules are applicable to insure a viable web configuration. They are as follows:

1. All Processor objects must meet the requirements of a generic processor object model. The fundamental requirements of such a generic processor is that the Processor has 0 or more input pins, 0 or more output pins and 0 or more update pins. Obviously, in order to be of any use, any given Processor object must have at least 1 input and at least one output pin of some particular type.
2. Input and output pins on each object function to inter-connect processor objects. The input pin for a processor object indicates which type of Result object(s) the Processor can consume, whereas, the output pin for a processor object indicates which type of Result object(s) the Processor can produce. Therefore, for 'Processor B' to consume the results produced by 'Processor A', the input pin of 'Processor B' must be connected to the output pin of 'Processor A'. Update pins are used for synchronizing a processor's output result(s) with respect to its input result(s). As mentioned above, different types of processor objects will have a different number and type of pins, as is required for implementing its particular functionality.

Fig. 2 (Examples of Processor classes) illustrates the application of this generic model to several specific processing classes that are typically encountered in digital oscilloscopes. These processor functions are merely representative of a very large number of processor functions that

might be implemented or defined by a user in a processing web in accordance with the Invention. In practice, these processor functions may be employed into a processing web by placing the element on an appropriate screen, and connecting the various inputs and outputs to implement the processing functions as desired.

In Fig. 2, an Acquisition Board processor class 210 does not have any inputs 216 and has 4 outputs 216 that produce waveforms (1 for each acquisition channel on the board, i.e. C1, C2, C3 and C4 of Fig. 1). A Waveform Averager processor class 220 has 1 input 222 that comprises an input waveform and 1 output 226 that produces a waveform. Furthermore, it includes an update pin 224 that explicitly controls when the waveform produced from the output is updated with respect to the waveforms seen at the input (i.e., precisely the timing of when the processing in the waveform averager is to be implemented). A Waveform Adder processor class 230 has two inputs 232 that comprise waveforms and one output 236 that produces a waveform. Waveform Adder processor class 230 does not have any update pin, and therefore, the waveform produced by its output is always continuously updated in real time with respect to the waveforms seen at its inputs. A Trace Renderer processor class 240 has 1 input 242 that comprises a waveform and 1 update pin 244 that specifies explicitly when the Trace Renderer should sample the waveforms seen at its input. It does not have any result outputs, but rather instead of producing any further results, it draws in a graphics window a representation of the waveforms seen at its input. The final two examples shown in Fig. 2 illustrate other types of results (namely parameters and histograms). An Amplitude processor class 250 comprises waveforms from its input 252 and produces parameter results at its output 256. A Parameter Histogrammer processor class 260 comprises parameter results at its input 262 and produces histogram results at its output 266, with an update pin 264 resetting the histogram and beginning the data

accumulation process. Thus, because it has an update pin, it is explicitly controlled via the update pin to know when it should receive parameter results at its input and update the histogram results produced at its output. A further description of update pins and the control thereof will be discussed below.

Fig. 2 only touches on a few simple examples of the types of processing classes that may be useful in an oscilloscope or other digital signal processing apparatus. As noted above, the design in accordance with the invention allows that any particular processor class may have as many input, output and update pins as is necessary or convenient to implement a particular processing functionality for that processor class. Furthermore, the result types required for any input or generated by any output pin may vary. For example, it would be possible to implement a processor class as shown in Fig. 3, where an example of a processor with multiple result types consumed and produced is shown. In this example, an 'Advanced Processor' class 310 consumes waveforms on its first input 312, and parameters on its second input 313. It produces results from four outputs, waveforms from the first output 316, histograms from the second output 317 and parameter results from the other two outputs 318, 319.

As noted above, the input and output pins for a particular processor class provide the ability to inter-connect processor objects, so that one processor object may consume the results produced from another processor object.

The update pin provides an explicit mechanism for specifying when the processor should consume results from its input(s) and in turn update the results that it produces from its output(s) based upon the newly consumed inputs. Only certain types of processors are provided with update pins. Processors that are cumulative in nature are the most common examples that include update pins. The term cumulative indicates that the processor accumulates information from each

result that it consumes from its input over a predefined time period and somehow merges that information together with previously-accumulated information to produce its output result(s). A processor that performs averaging of many input results is an example of a processor with an update pin. When instructed to update, the averager will consume as many results from its input as are available, accumulate data from each, and then average by dividing by the total number of results represented in the accumulated data to produce its output result. The output result will then remain in that state producing the same output result until the next time that the averager is instructed to update, at which point it will repeat its update procedure and produce a new output result. Therefore, because most processor classes do not include use of this explicit update mechanism, they do not have any update pins. This is because most processors produce their output result(s) as a function of the result(s) currently available at their input(s) at the time when their output is being requested.

That being said, in any given processing web configuration, there must always be at least one processor object with an update pin in order for any processing work to be performed. Fig. 4 (View Filter of C1 Processing Web Configuration) depicts a simple processing web configuration where a Trace Renderer 430 views an FIR-filtered version of an Acquisition Board's 410 C1 output 416. No processing work is performed in this configuration until the Trace Renderer 430 is updated via its update pin 434. At this time, the Trace Renderer 430 will request results from the output 426 of the FIR Filter 420, which will in turn request results from the C1 output 416 of the Acquisition Board 410. Thus, the implementation of an update pin for a particular element begins a procedure whereby requests for information are passed upstream until the requested data is available. When the Trace Renderer 430 finishes processing the

results (i.e. building a displayable image of those results), it will release the results and the configuration will become idle again until the next update cycle of the Trace Renderer.

FIG. 5 (View Average of C1 Processing Web Configuration) shows a slightly more complicated configuration, in which we see two processor objects with update pins, a Waveform Averager 520 and a Trace Renderer 530. In this scenario, the configuration is idle until either Waveform Averager 520 or the Trace Renderer 530 are updated via their update pins 524, 534, respectively. When Waveform Averager 520 is updated, it will request results from Acquisition Board's 510 C1 output 516 and will accumulate and normalize those results to produce a new output result 526. When Trace Renderer 530 is updated, it will request results from Waveform Averager's 520 output 526 and will process those results by building a displayable image. If Trace Renderer 530 is updated again before Waveform Averager 520 has been updated, then it will simply get the same result from the Waveform Averager's output. In this case, Trace Renderer would not have any processing work to perform unless some aspect of its definition had changed such that it were necessary to re-process the same data (e.g. graphical window-size changed and displayable representation was not re-scaleable). Thus, proper synchronization of the various update pins and data transfer sequences is essential, especially if it is desired to run various processor objects at different clock speeds, as will be discussed below.

As is clear from this description, it is necessary to manage the implementation of the various update pints to insure that proper data is received at each node in the system and that the sequence of update pin implementation insures proper functioning of the system. A Processing Web Manager software object is therefore responsible for managing the Processing Web and updating subsets of Processor objects in response to requests and events from other software objects. This Processing Web Manager object maintains a list of all of the processor objects and

analyzes the inter-connection paths and processor objects themselves to determine which processors should be updated and in which order based upon which events. In order to determine when the processors should be updated, the Processing Web Manager object reacts to 3 different events or requests: (1) an indication that new results are available at an output of some processor object (New Results Available), (2) a request for a synchronization of all or some of the processors on the processing web such as when a particular function (SynchronizeWeb) is to be performed that would require reprocessing of data by a number of processing objects and (3) a change in the definition of one of the processing objects on the processing web (Definition Changed). The first two events (1) New Results Available and (2) Synchronize Web) are posted by other fundamental system manager objects. A Result Source Manager (or Acquisition Manager) tells the Processing Web Manager when new source (acquisition) results are available via the NewResultsAvailable request. A Result Sink Manager (or Display Manager) tells the Processing Web Manager when it wishes to take a 'synchronized snapshot' of the renderer (or result sink) processor objects via the SynchronizeWeb request. Such a snapshot may be employed upon a downstream request for data, resulting in clocking through of upstream data. The DefinitionChanged request is used by any of the included processor objects to notify the Processing Web Manager when a processor's definition has changed in order to update all processing objects in response to the processor's definition change.

The Processing Web Manager divides the processors that should be updated into two subsets. The first set of processors is updated in response to NewResultsAvailable requests. The second set of processors is updated in response to SynchronizeWeb requests. By making these input requests completely independent of each other, it becomes possible to support two different

update rates for the two subsets of processors. In fact, in an oscilloscope this becomes an important feature particularly beneficial at high speed and performance, high data-throughput operating points, where the acquisition system can generate waveform results at a much higher rate than the display system can consume them. In such cases, if the processing web configuration contains other cumulative processor objects, then these can be utilized to accumulate many acquisition (source) results between each display system request for synchronization. It is only this inventive functionality that allows for this beneficial procedure.

FIG. 6 (Update Sequence: View Average of C1 Processing Web Configuration) illustrates an example of a possible sequence that takes advantage of this separation of processing paths. The numbered arrows 1-2-3 indicate the acquisition-driven thread of execution and the sequence thereof. The lettered arrows indicate the display-driven thread of execution and the sequence thereof. These two threads of execution are independent of each other as the Processing Web Manager updates separate processors in response to each event noted above. In high data-throughput cases, the numbered 1-2-3 sequence may execute many, many times for each time that the lettered A-B-C sequence executes thus providing better measurements in a shorter amount of time. Thus, while acquisition may take place at a very high rate, display may take place at a slower rate, thus generating a fast path/slow path combination.

However, when two such data rates are employed, precise synchronization is very important. The example of Fig. 6 will be used to explain this requirement. The waveform averager will acquire signals at a very high rate, and output an average waveform. The trace renderer will display this output average waveform. If not properly synchronized, the waveform averager might be showing a result from a set of waveforms not including the current waveform, or including waveforms after the display of the current waveform, rather than showing the

average of all waveforms up to and precisely including the current waveform displayed. Thus, it is imperative that, for example, the trace renderer be updated at a precise state of the faster updating waveform averager to give proper synchronized results.

This update procedure results in the added benefit of saving processing time. For example, the trace renderer does not continually inquire whether there is more data to display. Rather it requests to receive data from the waveform averager to display when updated. Thus, the processing object is only active when it is to perform a function.

This therefore generates a "pull-type" system where a downstream element requests data from an upstream element, and only then is data forwarded. An added benefit of this type of processing is that no intermediate buffers are required. Because data only arrives after requested, no storage is necessary. Thus, this benefit of the elimination of intermediate buffers is achieved, even with a great reduction in processing requirement, as noted above.

Another benefit of this "pull-type" system is that the number of processed results are not necessarily constrained to a one-to-one relationship with the input sources. In older oscilloscopes, if one waveform was input to a processing element, one waveform resulted after processing. However, in accordance with the invention, any number of inputs or measurements may generate any number of outputs. Thus, zero, one or more outputs may be generated for any number of inputs. Most older oscilloscopes can perform, for example, processing on a single waveform. However, as a simple example, in accordance with the invention, two waveforms may be input, and an average waveform be output, or a plurality of waveforms may be input, and a maximum and minimum average waveform may be output. Any other function, may be performed, using any number (M) of inputs and outputs (N) as desired.

Because of the ability to allow for the stringing of a multiple processing elements as desired, a graphical representation of such a processing system is even more necessary. Thus as is shown, any number of processing elements can be placed onto the processing web. Any object can be viewed. Basically there is no limit to the complexity of a given "processing web" permitted within the basic architecture. This is not to say that excessive flexibility is something desirable in a basic lab instrument. However in a special purpose application, it is highly desirable that the architecture is unconstrained.

As is therefore shown in Figs. 7A and 7B, different configurations of more complicated processing webs are displayed. Fig. 7A depicts a fixed web, but one that includes far more complexity than the processing web shown in Fig. 1. As is shown in Fig. 7A, plug-in probes or the like A and B are provided for. These plug-in elements each produce four input waveforms A1-A4 and B1-B4 respectively. Also provided are eight static memory locations M1-M8 for storing eight different static system inputs. These inputs any comprise any type of desired data necessary to implement any of the desired available processing functions. Also provided are up to 20 parameter measurements that may be displayed upon determination, or used as inputs for various processor functions

The processing functions available in this web will be described next. As is shown, eight traces are available, each employing two functions. Thus, each function displayed within the web includes a corresponding function along the outside of the web for display. As is further shown, each function within the web is able to accept two inputs. Each function along the outer edge of the web may similarly accept two inputs, but one of these inputs is from the corresponding function from within the web. Thus, for each displayed trace, two functions may be applied to a total of three inputs. Thus, while some flexibility is available in such an

apparatus, complete flexibility is not available. Additional chained functions cannot be employed, and additional inputs may not be provided. Thus, the functionality of the actual processing is somewhat limited.

Fig. 7B displays a fully configurable processing web, and is most representative of the ability and features of the present invention. Four dynamic channels 710, and static memory data types 720 are shown. Of course, any number of these dynamic or static objects may be provided. As is also shown four views 730 are also shown. These views 730 display the dynamic input information. Trace views 740 and parameters are 750 also provided. However rather than the input to these trace views comprising only dynamic inputs 710, static memory inputs 720 and parameter inputs 750, any number of processing objects 760 may be provided for performing various desired processing before output data is forwarded to trace views 740. Thus, in opposition to Fig. 7A in which a fixed structure must be followed, in the system displayed in Fig. 7B, and configuration of processing is possible. Multiple processors may be chained together, utilizing multiple inputs and producing multiple outputs. Processors may be used to produce data for more than one further processor, etc. There is virtually no limit to the complexity or configuration of processing that may be performed in the system. Therefore, in accordance with the inventive depiction of this graphical processing system, it is the graphical depiction of this processing web, along with a method for modifying the functional interrelationships within the processing web (discussed below) that allow for the precise functionality embodied in this invention.

This new graphical representation and new architecture provides complete flexibility in terms of how many processing elements compose a given processing web and opens the field to handle other more specific types of processing engines and their results.

In a preferred embodiment of the invention, the kinds of processing results that may be output from a processing object and therefore is eligible as an input to another processing element may comprise:

1. Waveforms (stored in either floating point precision or fixed point precision)
2. Parametric results that are not single valued, and which carry horizontal information as well as parametric information ... e.g. risetime estimate, accompanied by "when" the risetime occurred relative to the trigger or zero horizontal reference.
3. XY associations or pairs of associated values, accompanied by their roughly common horizontal information.
4. Complex waveforms consisting of their real and imaginary components.
5. Persistence "maps" or two-dimensional histograms.
6. One-dimensional histograms.
7. Boolean results of pass/fail etc. or other such processes.
8. Table results for collections of parametric results.
9. Cursor results that are graphical in nature ... a complementary aspect of parametric results ... and special in the sense that their graphical nature cannot be other than observed ... one cannot calculate on them.

A Processing Web Editor (PWEditor) constructed in accordance with the invention is a tool for presenting the configuration of the Processing Web (Web) of the invention, enabling the processing web to be reconfigured, and enabling the properties of the various elements in the web to be viewed and/or modified. As noted above, the inventive Processing Web defines the flow of data from the input of a DSO through various stages of processing to the display device.

The PWEditor is the most natural way to view and reconfigure the processing web, allowing for a true graphical representation of the processing web, and the ability to modify it as desired. The Processing Web consists of various types of Processing Nodes or objects. These are classified into various types based upon the number, and data type, of the various inputs and outputs (pins) associated therewith. This classification allows functionality provided by traditional DSOs to be described in the processing web, although the web is flexible enough to support non-traditional processing, i.e. nodes which produce simultaneously both waveforms and parameters. As partially noted above, these nodes may comprise:

1. Acquisition Systems, which are responsible for the presentation of (primarily) waveform data produced by the acquisition hardware in the DSO.
2. Math nodes, which accept waveform data and produce waveform data.
3. Parameter nodes, which accept waveform data and produce scalar (parameter) results.
4. Pass Fail nodes, which accept waveform and/or parameter inputs and produce a Boolean (pass/fail) result.
5. Adapter nodes, which provide data-type conversion, i.e. float -> integer waveform (vector) types.
6. Renderer nodes, which accept any data type and produce graphical results (waveform traces, parameter readouts, etc).

The various input and output pins of the processing nodes are therefore classified based upon the data type that they expose/accept.

A description of an example of various of the functions that may be employed in the PWEditor to change the functional structure of the processing web, and therefore the actual functioning of the oscilloscope will now be described.

A first feature available with the PWEditor is a live preview feature. Editing a Processing Web using the PWEditor is more intuitive when the results of each processing step may be visualized 'live', directly in the editor. Therefore, at each node, an output valve is displayed on a miniature display at the location of the node in the graphical representation of the processing web. These miniature renditions of the results are updated in semi real-time and avoid the need to switch between the editor and the traditional DSO display in order to view the results for any particular node. They are therefore coupled to a desired pin of a processor and a result is viewed. In a preferred embodiment, these miniature renditions appear as shown in Fig. 8. They are applicable as labeled.

Another feature of the PWEditor is the ability to edit the properties of a particular processing node. For example, many of the processing nodes that may appear on a processing web require various properties (controls) to be defined. A right-click with the pointing device over any of the processing nodes or other method of selection presents a popup dialog that allows the node's properties to be viewed, and edited. Thus, all properties may be defined, and therefore comprise a nodes definition. Examples of the parameters that may be set for a histogram node are shown in Fig. 9. The user has the opportunity to choose the horizontal scale 910, vertical scale 920 and other defined information 930.

Another feature of the invention that allows for easy editing of the processing web is the automatic insertion of adapters. When an attempt to connect two Pins with differing types is made the Processing Web Editor can use an adapter from a collection of Adapters to provide an automatic conversion between the two types. For example, if a user tries to make a connection between two pins that require different formats, such as placing an integer output into an input

that requires a floating point valve the automatic insertion of integer->float waveform adapter may be employed.

Thus, as is shown by way of example only, attempts to drag a connection between 'C1' and 'In'. The C1 pin is an integer waveform type, the FFT input 'In' is a floating-point waveform type.

As is then shown in Fig. 10 B, the Connection is made using a normally hidden adapter without requiring any further input or assistance from the user.

Finally, in Fig. 10C, when the Processing Web Editor is configured to show all hidden adapters, the adapter for converting from the integer value to the floating point valve is shown.

As another example, a user may require converting between a waveform (vector) type valve and a parameter (scalar) type valve.

As is shown in Fig.11A an attempt is made to connect C1 output to a histogrammer's input. C1 is an integer waveform type, the histogram input is a scalar type.

As is then shown in Fig. 11B, the connection is made with hidden adapters inserted.

Finally, as is shown in Fig. 11C, if the PWEditor is instructed to display the hidden adapters, the complete picture of how the connection was made is displayed. Thus, first integer waveform data 110 is converted to floating-point waveform data at 120, then a second adapter converts 130 the floating-point waveform data into multiple parameter values (scalars) which may then be fed to the histogrammer 1140 to generate a histogram output 1150.

These various connection features and processing elements may be employed through a graphical user interface of the PWEditor. Features thereof will now continue to be described.

Processing engines may be 'dragged' from a categorized palette onto the web. Thus, if a user wishes to perform a particular processing the appropriate processing function need only be

dragged into the processing web. Once present on the web, the inputs and outputs of the selected processing function may be connected by dragging from output to input of the various inputs and outputs of other objects on the web using the pointing device.

Fig. 12 shows an example of a collection of components in the Math category. Palettes for other categories may be provided, as noted on the various tabs.

An example of designing a simple processing web in accordance with the invention will now be described. Fig. 13 shows one of the simplest forms of a processing web. This shows the Rendering of four waveforms. This would be the state of the processing web if channels 1 through 4 in a traditional DSO are turned on. In this figure, a user has selected a waveform generator 1310 and four display elements 1330. These five elements were dragged onto the web area. And each output from the waveform generator was connected to the input of one display element. This configuration would correspond to the portion of the graphical representation of the processing web of Fig. 1, including only nodes 110 ($C_1 - C_4$) and 130 ($C_1$ view – $C_4$ view) and direct connections therebetween.

Fig. 14 depicts an example is similar to Fig. 13 above, but adds a parameter node 1450 (a waveform amplitude calculator), and shows it's result at 1460.

Thus, in addition to providing the five objects of Fig. 13, the waveform amplitude calculator 1450 and display thereof 1460 were dragged into the processing web, and connected as shown in Fig 14. Thus, the parameter measurement is performed at 1450, and the numeric parameter output is displayed at 1460. The output form node 1110($C_1$) is forwarded to two locations, one of display (1330) and one for further processing (1450, 1460).

The example processing web shown in Fig. 15 builds upon the web displayed in Fig. 14, and adds a histogrammer 1570 to show the distribution of the amplitude parameter result 1580.

Thus the output from the output from the parameter mode 1450 is forwarded to histogramer 1570 in addition to parameter display 1460. Histogrammer 1570 calculates the distribution of the value at parameter node 1450, and displays the histogram result at display 1580. It is this ability to easily add various processing and viewing functions that in part define the invention.

Referring next to Fig. 16, the method by which the processing web editor screen display is populated in accordance with an existing processing web will be described. At a first step 1605 after the starting of the procedure, the processing web editor asks the processing web manager for a first component comprising the processing web. Then, at step 1610, it is determined whether such a component has been provided. If such a component has been provided, control passes to step 1615 where the various details of the provided web component, such as pin details, icon and label are determined. Thereafter, at step 1620, an icon is placed on the processing web display at a position defined by the processing node's properties. These properties may comprise where in the proximity to the waveform generation elements, or the number of elements to be placed before after this particular component. Then, at step 1625, input and output pins are drawn, preferably in color based upon the data type to be output from or input thereto. Then, at step 1630, the processing web manager is asked for the next component in the web. Control then returns to inquiry 1610. If another component is available control then continues to pass to step 1615 and the procedure noted above is performed once again.

If however, at step 1610 no further components in the web are available, and therefore the inquiry is answered in the negative, control then passes to step 1635 where the processing web manager is asked for the first connection in the web. Control then passes to step 1640 where it is determined whether such a connection has been provided. If so, control passes to step 1645

where the pins defined by the connection are connected by a line, preferably colored based upon the data type flowing through the connection. Of course, any other graphical designation may be employed for a line, such as texture, crosshatch, a symbol or the like. After this connection has been graphically illustrated, control passes to step 1650 where the processing web manager is asked for a next connection in the web. If such a connection exists, as in step 1640, the inquiry will be answered in the affirmative and control will pass to step 1645, thereby repeating this procedure. If at step 1640 it is determined that no further connections in the web exists and the entire web has been graphically illustrated, the inquiry at step 1640 will be answered in a negative and the process will end. Therefore, in accordance with the procedure set forth in the phase 16, a graphical representation can be displayed to a user of an existing web.

Referring next to Figure 17, a process for dropping a new component into the processing web, as displayed by the processing web editor, will be described. Upon starting the procedure, at step 1710, a call is placed to the processing web manager with a request for adding a component to the processing web. In the next step 1720, the processing web manager calls a subroutine called "ProcessorAdded" to be acted upon in the editor, with a pointer passing a parameter indicating the new processor to be added. Thereafter, in the next step 1730, an icon corresponding to the processor to be added is generated at a position to find by the processing properties in a matter noted in Fig. 16. Then, at step 1740, input and output pins are drawn, preferably in a color based upon the data type input or output therefrom, and at step 1750, pins are connected by a connection of the new component with the previously existing components. The lines for the connections are drawn preferably in color based upon the type of data flowing through the connection as noted with respect to Fig. 16. After these connections are made, the component has been added to the processing web, and the procedure ends.

Referring next to Fig. 18, a procedure for attempting to connect two pins that are in different components by dragging a line between them with a pointer device will be described. Upon start of the procedure, at step 1810 an output to determined the type of source and destination pins. An inquiry at step 1820 asks whether the data types of the pins to be connected are the same. If this inquiry is answered in the affirmative, and the pins are the same type, control passes to step 1830 where the processing web manager is asked to directly connect the source to the destination pin. Thus, the graphical representation of adding a connection between these two pins also results in the modification of the actual processing web, and therefore the functioning of the data processing apparatus to be modified. Thereafter, the procedure will end.

However, if at step 1820 it is determined that the data types for the source and destination pins are not of the same type, at step 1840 it is determined whether an adapter is available that can convert these source type into the destination type. If this inquiry at step 1840 is entered into the affirmative, and an appropriate adapter is available, control passes to step 1850 where the processing web manager is asked to create the adapter. Thereafter, at step 1860 the processing web manager is asked to connect the source pin to the adapter's input pin and the adapter's output pin to the destination pin. After the conclusion of these connections, processing ends. Of course, while the addition of only one adapter is described, as noted above, multiple adapters may be provided in sequence. Finally if at step 1840 it is determined that an appropriate adapter is not available, control passes to 1870 and the connection is refused to be made. Processing then ends.

Therefore, in accordance with the invention, a highly configurable system is provided for acting upon signals in an oscilloscope or other data processing device.

Utilizing the inventive PWEditor to implement the inventive processing web provides the following unique features. The Complexity of the processing web is limited only by available processing power and memory. The traditional separation of Math, Parameters, Display functions is removed. Processing engines may be dragged into use from a categorized toolbar. An arbitrary number of named and typed input and output pins may be employed. Graphical, real-time, preview of results (both scalar and vector) are provided, even while viewing the web structure. Identification of the data-type of input and output pins may be designated using a color-coded key scheme. Data type adapters are automatically inserted when incompatible types are connected, for example, adaptation between float and integer data types, conversion of vector data into a sequence of scalars, etc. There is also the ability to display, and edit the properties of each node and each pin.

In accordance with the PWEditor constructed in accordance with the invention, any number of processing elements can be placed onto the processing web. Any object can be viewed at any time, in real time. There is no theoretical limit to the complexity of a given processing web permitted within the basic architecture. While excessive flexibility may not be desirable in a basic lab instrument, in a special purpose application, it is highly desirable that the architecture be unconstrained and modifiable to a high level of complexity. This new architecture in accordance with the invention provides complete flexibility in terms of how many processing elements compose a given processing web, how a user is able to view and even conceptualize this processing web, and opens the field to handle other more specific types of processing engines and their results.

Thus, as noted above, the PWEditor may be used to show and configure the actual existing processing engines in a network which represents in more detail what happens in the

processing part of the conceptual model for an oscilloscope. The actual network of processing elements is, as noted above, referred to as the "processing web". This PWEditor for viewing and configuring this processing web therefore has several innovative features.

1. It shows iconically the types of engine and with color the types of inputs and type of outputs supported and provided by the engine. In a preferred embodiment of the invention, each processor's inputs and outputs are color coded based upon the types of inputs that are required and the types of outputs that are provided. Of course, as noted above, any other type of designation may be employed. Thus by connecting outputs and inputs of different processing elements that are of the same color, proper types of data being transferred can be insured.
2. It permits adding objects, by drag-and-drop from various toolbars, onto the processing web, and attaching the input "pins" and output pins of the objects concerned.
3. Acquisition systems are shown as objects with (typically) only output pins (although one could imagine calibration signals being shown as inputs)
4. Displays (or views or renderings) of any "pin" in the processing web is possible, and is shown in miniature ... "live". (Meaning there is a live display of the pin)

As is therefore shown in Figs. 19 and 20, views of examples of functioning PWEditors are shown that have been designed and populated according to the disclosures above. Various viewing and implementation instructions 1910 are included along the top row while below them are a list of categories 1915 of different functions a user might want to implement in the processing web. The main window portion 1900 of the PWEditor includes the various processing functions 1920 that have already been selected by a user, with the interconnections

1925 therebetween shown. Also shown are various output displays 1930 for displaying a current value of the web at certain locations.

This displayed processing web was built in accordance with the PWEditor in accordance with the invention as described below. When using the PWEditor, any object in the processing web (including renderers and acquisition systems) can be selected by clicking on it with a pointing device, or implementing the same pointing functions employing a touch screen. Once selected, the properties of the object (e.g. setup of the acquisition system or colors of the view) can be seen and modified by touching on the properties of button or alternatively by using a right click with a more traditional pointing device. The result types are preferably color-coded, the various colors for the input and output pins being noted in accordance with a color key provided in the lower left of the display.

Each output and input "pin" of each processing component is shown with its name and is also preferably shown in the color, or other designation of its corresponding result type. Input pins are neither limited in number nor in type. Any and all combinations are permitted.

Each element shown in Figs. 19 and 20 represents a real "component" in the actual system controlled in accordance with the processing web designated by the PWEditor. In this design the formalism of COM is used (the component object model) as supported by Microsoft corporation. This permits a coupling of these components that can be accomplished very late, even at run time.

The executable (binary) code which performs the actual calculations are not necessarily loaded into the computer's memory before the operator (human or automated) actually places an object onto the processing web. In this sense the application can be running before the code in question is even written or compiled. Therefore, software supporting probes, or processing

related to probes or other physical data input devices will be delivered in the physical hardware of the probe or other physical data input device. This has a commercial advantage in the marketplace since solutions can be delivered in an evolutionary way.

It will thus be seen that the objects set forth above, among those made apparent from the preceding description, are efficiently attained and, because certain changes may be made in carrying out the above method and in the construction(s) set forth without departing from the spirit and scope of the invention, it is intended that all matter contained in the above description and shown in the accompanying drawings shall be interpreted as illustrative and not in a limiting sense.

It is also to be understood that the following claims are intended to cover all of the generic and specific features of the invention herein described and all statements of the scope of the invention which, as a matter of language, might be said to fall therebetween.